Software for verification of collision avoidance algorithms via Optimal Control Techniques

Ilaria Xausa, Robert Baier, Olivier Bokanowski, Matthias Gerdts, Daniel Toepfer

VOLKSWAGEN

AKTIENGESELLSCHAFT

ITN SADCO
Initial Training Network
Sensitivity Analysis for Deterministic Controller Design

Software for verification of collision avoidance algorithms via Optimal Control Techniques.

Ilaria Xausa - Joint work with Robert Baier, Olivier Bokanowski, Matthias Gerdts and Daniel Toepfer

Conference on New Trends in Optimal Control - Tours, June 23-27 2014

Contents

Motivation

Goal

Virtual Test Maker
- Input interface
- The optimal control problem
- Answers to VW questions

Numerical simulations
- Comparison with CAB
- Comparison with collision by steering algorithm

Motivation

- Modern driver assistance systems require high standard security qualifications before entering the market.

Motivation

- Modern driver assistance systems require high standard security qualifications before entering the market.
- Important is to find mistakes in the software packages considering the sensor accuracy.

Figure : Several sensors to detect scenario (left). Warning of Advanced Driver Assistance Systems (ADAS) (right).

Figure : ADAS algorithm (left) can give a false warning detected by tests on road (right).

Motivation

- Modern driver assistance systems require high standard security qualifications before entering the market.
- Important is to find mistakes in the software packages considering the sensor accuracy.
- Currently, this requires exhaustive tests in real world scenarios, demanding high costs.

Figure : Several sensors to detect scenario (left). Warning of Advanced Driver Assistance Systems (ADAS) (right).

Figure : ADAS algorithm (left) can give a false warning detected by tests on road (right).

Motivation

- Modern driver assistance systems require high standard security qualifications before entering the market.
- Important is to find mistakes in the software packages considering the sensor accuracy.
- Currently, this requires exhaustive tests in real world scenarios, demanding high costs.
- Release testing of todays ADAS requires up to 2 million test km and 1.000 test drivers.

Figure : Several sensors to detect scenario (left). Warning of Advanced Driver Assistance Systems (ADAS) (right).

Figure : ADAS algorithm (left) can give a false warning detected by tests on road (right).

Goal

- We create a virtual test maker to shift the test from the street to simulation by using different optimal control techniques.
- Here, we focus on validating Collision Avoidance systems using Steering (CAS) and Braking (CAB) maneuvers. In particular, we investigate the activation times, which may be too early or too late due to sensor errors or algorithmic errors.
- We validate ADAS algorithms for driver assistance systems in collision avoidance (collision avoidance by braking and steering) through Virtual Test Maker (VTM).

What does “validate” mean?

	On road	On software basis
Test	Drive many different scenarios and detect environment through sensors affected by errors	Tests on different scenarios
Collect data	Collect **driven** data	Collect **simulated** data
Compare	Among all the data how do we define a too early/late activation of ADAS warning or maneuver?	**Interpret the data** to define when and why is ADAS algorithm giving a too early/late activation warning or maneuver
Evaluate	**How can ADAS be improved?** Look at specific scenarios where the activation of ADAS warning or maneuver is too early/late. **How errors in sensor data** affect a collision avoidance perspective.	

What does “validate” mean?

	On road	On software basis
Test	Drive many different scenarios and detect environment through sensors affected by errors	Tests on different scenarios
Collect data	Collect **driven** data	Collect **simulated** data
Compare	Among all the data how do we define a too early/late activation of ADAS warning or maneuver?	**Interpret the data** to define when and why is ADAS algorithm giving a too early/late activation warning or maneuver
Evaluate	**How can ADAS be improved?** Look at specific scenarios where the activation of ADAS warning or maneuver is too early/late. **How errors in sensor data** affect a collision avoidance perspective.	

Our software aims to substitute **tests on road**

What does “validate” mean?

	On road	On software basis
Test	Drive many different scenarios and detect environment through sensors affected by errors	Simulations for different scenarios
Collect data	Collect **driven** data	Collect **simulated** data
Compare	Among all the data how do we define a too early/late activation of ADAS warning or maneuver?	**Interpret the data** to define when and why is ADAS algorithm giving a too early/late activation warning or maneuver
Evaluate	**How can ADAS be improved?** Look at specific scenarios where the activation of ADAS warning or maneuver is too early/late. **How errors in sensor data** affect a collision avoidance perspective.	

Our software aims to substitute **tests on road** with **virtual tests**

What does “validate” mean?

	On road	On software basis
Test	Drive many different scenarios and detect environment through sensors affected by errors	Simulations for different scenarios
Collect data	Collect **driven** data	Collect **simulated** data
Compare	Among all the data how do we define a too early/late activation of ADAS warning or maneuver?	**Interpret the data** to define when and why is ADAS algorithm giving a too early/late activation warning or maneuver
Evaluate	How can ADAS algorithms be improved? Look at specific scenarios where the activation of ADAS warning or maneuver is too early/late. How errors in sensor data affect a collision avoidance perspective.	

Our software aims to substitute **tests on road** with **virtual tests** to answer the unique **evaluation question**.

Virtual Test Maker

INPUT INTERFACE

- Specify validation goals
- Specify scenario

→

Model as **O**ptimal **C**ontrol **P**roblem

→

Software for solving optimal control problems

- OCPIDDAE
- ROC-HJ

→

OUTPUT INTERFACE

- Data files
- Matlab plots

Virtual Test Maker: input interface

INPUT INTERFACE
- Specify validation goals
- Specify scenario

→

Model as **O**ptimal **C**ontrol **P**roblem

→

Software for solving optimal control problems
- OCPIDDAE
- ROC-HJ

→

OUTPUT INTERFACE
- Data files
- Matlab plots

Virtual Test Maker: Specification - Validation Goals

Capabilities of VTM

(P1) Compute an optimal trajectory for avoiding a collision (optimal in the sense that is the fastest or the closest to the obstacle) in a particular scenario

(P2) Compute the set of all initial points from which it is possible to avoid a collision in a particular scenario

(P3) Compute the set of all end points that the car can reach from a given initial point in a particular scenario

Verification part

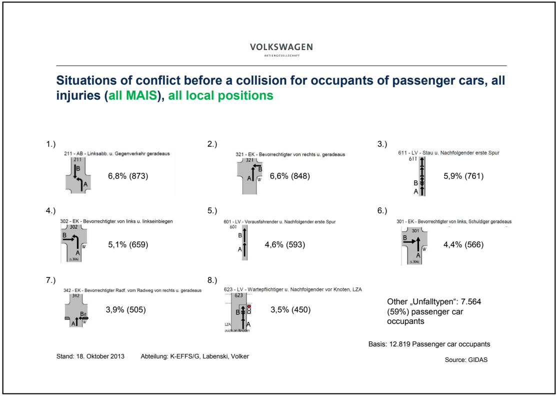
(V1) Validate collision avoidance by braking algorithm (CAB) for a test collision scenario

(V2) Validate collision avoidance by steering algorithm (CAS) for a test collision scenario

Virtual Test Maker: Specification - Scenario

Exemplary scenarios according to the VW database, collected by “GIDAS” (German In Depth investigation Accident Study)

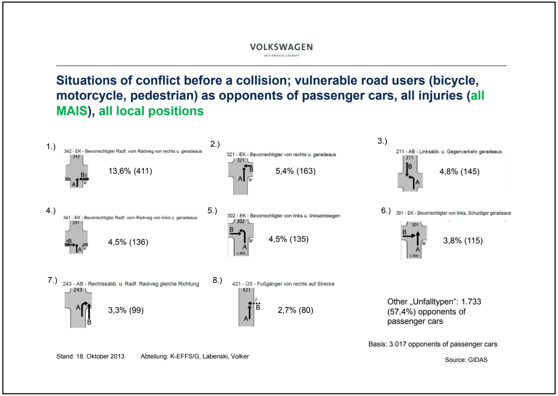
Most frequent scenarios considering all level of injuries (all MAIS) for occupants of car in both rural and urban roads (all local positions)



Virtual Test Maker: Specification - Scenario

Exemplary scenarios according to the VW database, collected by “GIDAS” (German In Depth investigation Accident Study)

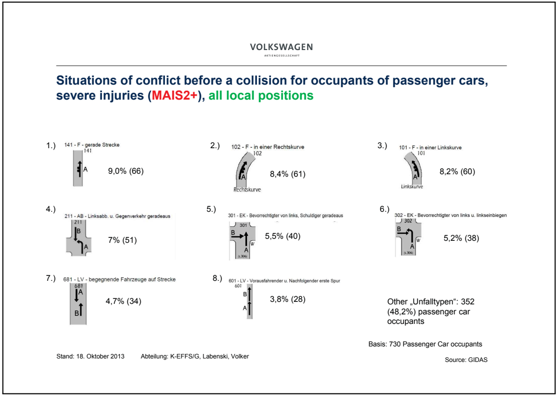
Most frequent scenarios considering all level of injuries (all MAIS) for occupants of obstacles in both rural and urban roads (all local positions)



Virtual Test Maker: Specification - Scenario

Exemplary scenarios according to the VW database, collected by “GIDAS” (German In Depth investigation Accident Study)

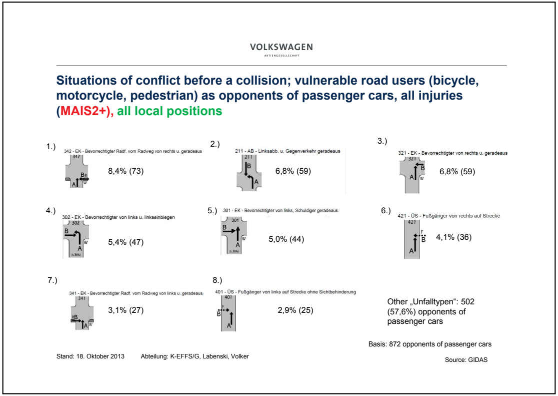
Most dangerous scenarios (injuries above level MAIS2) for occupants of car in both rural and urban roads (all local positions)



Virtual Test Maker: Specification - Scenario

Exemplary scenarios according to the VW database, collected by “GIDAS” (German In Depth investigation Accident Study)

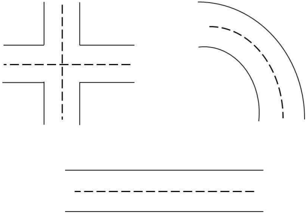
Most dangerous scenarios (injuries above level MAIS2) for occupants of obstacles in both rural and urban roads (all local positions)



Virtual Test Maker: Specification - Scenario

Exemplary scenarios consider crossing, curve, straight roads, with one or two players:

- road geometry $\rightarrow$ straight, curve, crossing $\rightarrow$ state constraints;



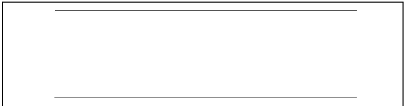
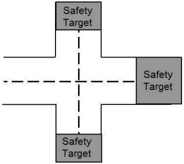


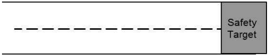
Figure : Example of scenario construction.

Virtual Test Maker: Specification - Scenario

Exemplary scenarios consider crossing, curve, straight roads, with one or two players:

- road geometry $\rightarrow$ straight, curve, crossing $\rightarrow$ state constraints;
- safety target (end of automatic maneuver) $\rightarrow$ function $\rightarrow$ boundary constraints;





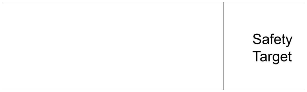
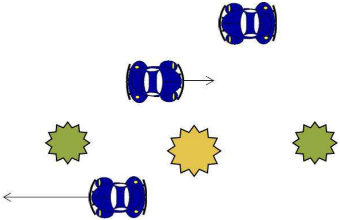


Figure : Example of scenario construction.

Virtual Test Maker: Specification - Scenario

Exemplary scenarios consider crossing, curve, straight roads, with one or two players:

- road geometry → straight, curve, crossing → state constraints;
- safety target (end of automatic maneuver) → function → boundary constraints;
- obstacles data → obstacle geometry (circle or rectangle), position, velocity → state constraints;



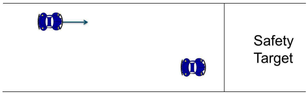


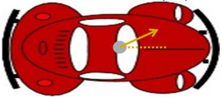
Figure : Example of scenario construction.

Virtual Test Maker: Specification - Scenario

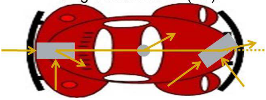
Exemplary scenarios consider crossing, curve, straight roads, with one or two players:

- road geometry → straight, curve, crossing → state constraints;
- safety target (end of automatic maneuver) → function → boundary constraints;
- obstacles data → obstacle geometry (circle or rectangle), position, velocity → state constraints;
- vehicle model → single track or point mass → states, controls, dynamics;

Point mass model (4D):



Single track model (7D):



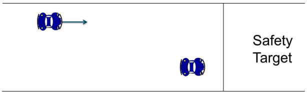
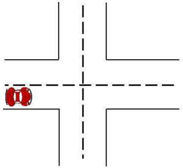


Figure : Example of scenario construction.

Virtual Test Maker: Specification - Scenario

Exemplary scenarios consider crossing, curve, straight roads, with one or two players:

- road geometry $\rightarrow$ straight, curve, crossing $\rightarrow$ state constraints;
- safety target (end of automatic maneuver) $\rightarrow$ function $\rightarrow$ boundary constraints;
- obstacles data $\rightarrow$ obstacle geometry (circle or rectangle), position, velocity $\rightarrow$ state constraints;
- vehicle model $\rightarrow$ single track or point mass $\rightarrow$ states, controls, dynamics;
- initial vehicle position $\rightarrow$ initial state (position, velocity, steering, yaw angle) $\rightarrow$ initial conditions;



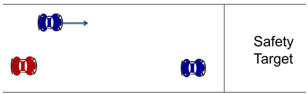
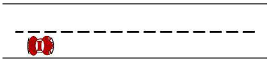


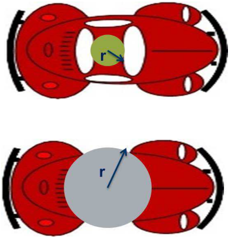
Figure : Example of scenario construction.



Virtual Test Maker: Specification - Scenario

Exemplary scenarios consider crossing, curve, straight roads, with one or two players:

- road geometry $\rightarrow$ straight, curve, crossing $\rightarrow$ state constraints;
- safety target (end of automatic maneuver) $\rightarrow$ function $\rightarrow$ boundary constraints;
- obstacles data $\rightarrow$ obstacle geometry (circle or rectangle), position, velocity $\rightarrow$ state constraints;
- vehicle model $\rightarrow$ single track or point mass $\rightarrow$ states, controls, dynamics;
- initial vehicle position $\rightarrow$ initial state (position, velocity, steering, yaw angle) $\rightarrow$ initial conditions;
- sensor accuracy $\rightarrow$ perturbation in the initial state (position, velocity, steering, yaw angle) $\rightarrow$ sensitivity analysis.



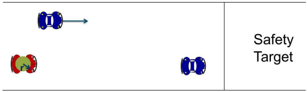


Figure : Example of scenario construction.

Virtual Test Maker: the optimal control problem (OCP)

INPUT INTERFACE

- Specify validation goals
- Specify scenario

→

Model as **O**ptimal **C**ontrol **P**roblem

→

Software for solving optimal control problems

- OCPIDDAE
- ROC-HJ

→

OUTPUT INTERFACE

- Data files
- Matlab plots

Virtual Test Maker: the OCP

- car model and initial states $\to$ states, controls and dynamics:
  Let $u \in L^\infty([0,T],\mathcal{U})$ a control policy with $T > 0$ and $\mathcal{U} \subset \mathbb{R}^m$ non-empty and compact, and z solution of the dynamics

$$z'(t) = f(z,u,t) \text{ a.e. } t \in [0,T], z(0) = z_0 \in \mathbb{R}^n.$$

4D Point Mass Model	$z = (x, y, \psi, v),$	$u = (w_\psi, F_B).$
7D-1 Single Track Model	$z = (x, y, \psi, w_\psi, v_x, v_y, \delta),$	$u = (w_\delta, F_B).$
7D-2 Single Track Model	$z = (x, y, v, \psi, w_\psi, \alpha, \delta),$	$u = (w_\delta, F_B).$

Virtual Test Maker: the OCP

- car model and initial states $\rightarrow$ states, controls and dynamics:
  Let $u \in L^\infty([0,T],\mathcal{U})$ a control policy with $T > 0$ and $\mathcal{U} \subset \mathbb{R}^m$ non-empty and compact, and z solution of the dynamics

$$z'(t) = f(z,u,t) \text{ a.e. } t \in [0,T], z(0) = z_0 \in \mathbb{R}^n.$$

4D Point Mass Model	$z = (x, y, \psi, v),$	$u = (w_\psi, F_B).$
7D-1 Single Track Model	$z = (x, y, \psi, w_\psi, v_x, v_y, \delta),$	$u = (w_\delta, F_B).$
7D-2 Single Track Model	$z = (x, y, v, \psi, w_\psi, \alpha, \delta),$	$u = (w_\delta, F_B).$

- road geometry and obstacle geometry, position and motion $\rightarrow$ state constraints:

$$\forall t \in (0,T), z(t) \in \mathcal{K} \Leftrightarrow g(z(t)) \leq 0$$

Virtual Test Maker: the OCP

- car model and initial states $\rightarrow$ states, controls and dynamics:
  Let $u \in L^{\infty}([0,T],\mathcal{U})$ a control policy with $T > 0$ and $\mathcal{U} \subset \mathbb{R}^m$ non-empty and compact, and z solution of the dynamics

$$z'(t) = f(z,u,t) \text{ a.e. } t \in [0,T], z(0) = z_0 \in \mathbb{R}^n.$$

4D Point Mass Model	$z = (x, y, \psi, v)$,	$u = (w_\psi, F_B)$.
7D-1 Single Track Model	$z = (x, y, \psi, w_\psi, v_x, v_y, \delta)$,	$u = (w_\delta, F_B)$.
7D-2 Single Track Model	$z = (x, y, v, \psi, w_\psi, \alpha, \delta)$,	$u = (w_\delta, F_B)$.

- road geometry and obstacle geometry, position and motion $\rightarrow$ state constraints:

$$\forall t \in (0,T), z(t) \in \mathcal{K} \Leftrightarrow g(z(t)) \leq 0$$

- safety target $\rightarrow$ boundary constraints:

$$z(T) \in \Omega \Leftrightarrow \varphi(z(T)) \leq 0$$

Virtual Test Maker: the OCP

- car model and initial states $\rightarrow$ states, controls and dynamics:
  Let $u \in L^\infty([0,T],\mathcal{U})$ a control policy with $T > 0$ and $\mathcal{U} \subset \mathbb{R}^m$ non-empty and compact, and z solution of the dynamics

$$z'(t) = f(z,u,t) \text{ a.e. } t \in [0,T], z(0) = z_0(p), p \in \mathbb{R}^n \text{ parameter}.$$

4D Point Mass Model	$z = (x, y, \psi, v)$,	$u = (w_\psi, F_B)$.
7D-1 Single Track Model	$z = (x, y, \psi, w_\psi, v_x, v_y, \delta)$,	$u = (w_\delta, F_B)$.
7D-2 Single Track Model	$z = (x, y, v, \psi, w_\psi, \alpha, \delta)$,	$u = (w_\delta, F_B)$.

- road geometry and obstacle geometry, position and motion $\rightarrow$ state constraints:

$$\forall t \in (0,T), z(t) \in \mathcal{K} \Leftrightarrow g(z(t)) \leq 0$$

- safety target $\rightarrow$ boundary constraints:

$$z(T) \in \Omega \Leftrightarrow \varphi(z(T)) \leq 0$$

- sensor tolerance $\rightarrow$ sensitivity analysis:
  it investigates the dependence of the solution $\hat{z} := z(\hat{u},\hat{p})$ of the initial value problem on p for a fixed (optimal) control $\hat{u} = \hat{u}(\hat{p})$ and the nominal parameter $\hat{p}$. more

Virtual Test Maker: the OCP

- question:
  - compute an (optimal) trajectory to a secure target state: we need the objective function.
  - compute the reachable set from an initial state z_0

$$\begin{aligned} \mathcal{FR}(T, z_0) = \quad & \{z(T) \in \mathbb{R}^n \mid \text{ given } z_0 \in \mathbb{R}^n, \exists u \in \mathcal{U}, \\ & z(u, z_0)(t) \text{ is admissible for OCP, } \forall t \in [0, T]\} \end{aligned}$$

  - compute the backward oriented reachable set starting from a secure state

$$\mathcal{BR}(T, z_0) = \{z_0 \in \mathbb{R}^n \mid \exists u \in \mathcal{U},\ z(u, z_0)(t) \text{ is admissible for OCP, } \forall t \in [0, T]\}$$

  - often a projected reachable set is of interest, where $\pi(z(T))$ or $\pi(z(0))$ needs to be calculated with π being a projection from $\mathbb{R}^n$ to the $2D$ or $3D$ space.

The mathematical details are in:

- R. Baier, M. Gerdts, I. Xausa, *Approximation of Reachable Sets using Optimal Control Algorithms*, Numerical Algebra, Control and Optimization, 2013.
- I. Xausa, R. Baier, M. Gerdts, M. Gonter, C. Wegwerth, *Avoidance Trajectories for Driver Assistance Systems via Solvers for Optimal Control Problems*, Proceedings of the 20th International Symposium on Mathematical Theory of Networks and Systems, 2012.

Virtual Test Maker: software packages for solving OCP

INPUT INTERFACE

- Specify validation goals
- Specify scenario

→

Model as **O**ptimal **C**ontrol **P**roblem

→

Software for solving optimal control problems

- OCPIDDAE
- ROC-HJ

→

OUTPUT INTERFACE

- Data files
- Matlab plots

Virtual Test Maker: software packages for solving OCP

- **OCPID-DAE1** [M. Gerdts]
  It is designed for the numerical solution of optimal control problems (Fortran 90 interface).
  The solution of many OCPs provides boundary and interior points as well as the distance function for the reachable set.
  `http://www.optimal-control.de`
- **ROC-HJ** [O. Bokanowski, H. Zidani]
  It is designed for numerical solution of n-dimensional Hamilton-Jacobi-Bellman equations (C++ interface).
  The solution of PDE provides a level set representation of the reachable set.
  `http://uma.ensta-paristech.fr/var/files/ROC-HJ/`

Virtual Test Maker: answers

INPUT INTERFACE

- Specify validation goals
- Specify scenario

→

Model as **O**ptimal **C**ontrol **P**roblem

→

Software for solving optimal control problems

- OCPIDDAE
- ROC-HJ

→

OUTPUT INTERFACE

- Data files
- Matlab plots

Recall...

Capabilities of VTM

(P1) Compute an optimal trajectory for avoiding a collision by steering (optimal in the sense that is the fastest or the closest to the obstacle) in a particular scenario

(P2) Compute the set of all initial points from which it is possible to avoid a collision in a particular scenario

(P3) Compute the set of all end points that the car can reach from a given initial point in a particular scenario

Verification part

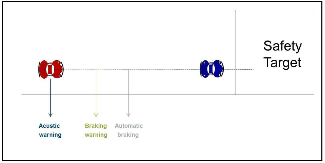
(V1) Validate collision avoidance by braking algorithm (CAB) for a test collision scenario

(V2) Validate collision avoidance by steering algorithm (CAS) for a test collision scenario

Comparison with CAB - Input of VTM

- a scenario (shape of road, obstacle positions and velocities, car position and velocity);
- output of CAB (three warning levels given as distance to the collision obstacle).

  - acoustical warning is given in distance to collision obstacle w_1;
  - brake warning is given in distance to collision obstacle w_2;
  - automatic braking is given in distance to collision obstacle w_3.



Comparison with CAB - Output of VTM

Warning level CAB	Virtual test maker output		
acoustic warning w_1	**last point to steer and last point to brake** with controls $(w_\delta, F_B) \in \mathcal{U}_1 \subset \mathcal{U}_2 \subset \mathcal{U}_3$	**optimal trajectory**, initial state is w_1, considering sensor errors	**reachable area**, initial state is w_1, considering sensor errors
braking warning w_2	**last point to steer and last point to brake** with controls $(w_\delta, F_B) \in \mathcal{U}_2 \subset \mathcal{U}_3$	**optimal trajectory**, initial state is w_2, considering sensor errors	**reachable area**, initial state is w_2, considering sensor errors
automatic braking w_3	**last point to steer and last point to brake** with controls $(w_\delta, F_B) \in \mathcal{U}_3$	**optimal trajectory**, initial state is w_3, considering sensor errors	**reachable area**, initial state is w_3, considering sensor errors

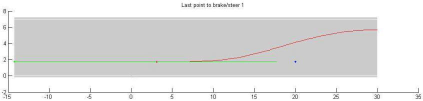
Comparison with collision by braking algorithm - Example of testing process

Input

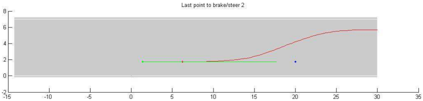
Output of VTM

- Last point to brake and last point to steer for each warning level (top to bottom):

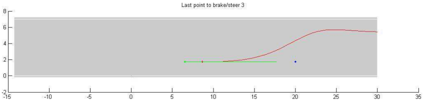
acoustic warning



braking warning



automatic braking

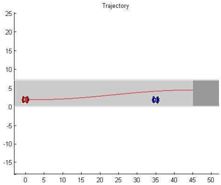


Comparison with collision by braking algorithm - Example of testing process

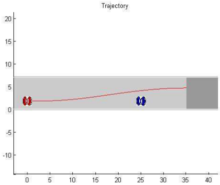
Output of VTM

- Avoidance trajectories for each warning level (left to right) (with perturbation):

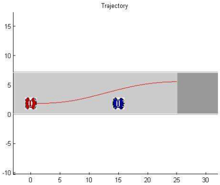
acoustic warning



braking warning



automatic braking

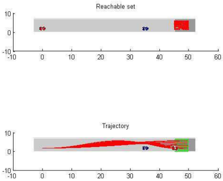


Comparison with collision by braking algorithm - Example of testing process

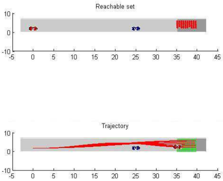
Output of VTM

- Reachable set, robust reachable set and trajectories to reachable set (top to bottom) for each warning level (left to right) (with perturbation):

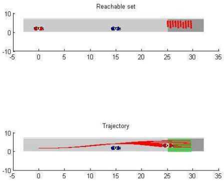
acoustic warning



braking warning



automatic braking



Comparison with collision by braking algorithm - Example of testing process

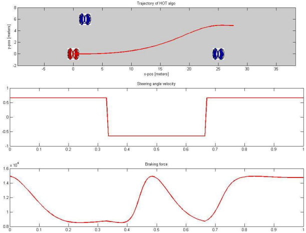
	Initial data			CAB warning distance		Virtual test maker	
	vel car [km/h]	vel obst [km/h]	acc obst [m/s^2]	upper bound [m]	lower bound [m]	lptb [m]	lpts [m]
w1	50	0	0	43.8889	32.222	34.229	16.880
w2	50	0	0	31.6667	20.5556	18.589	13.750
w3	50	0	0	19.4444	2.22222	13.462	11.342
w1	150	0	0	130	100	270.345	45.148
w2	150	0	0	98.3333	75	143.637	43.078
w3	150	0	0	73.3333	2.72853	100.415	36.325

Table : Summary of exemplary results.

Comparison with CAS - Input of VTM

The tool should provide:

- a scenario (shape of road, obstacle positions and velocities, car position and velocity);
- output of CAS (a collision avoidance trajectory is computed).



Comparison with CAS - Output of VTM

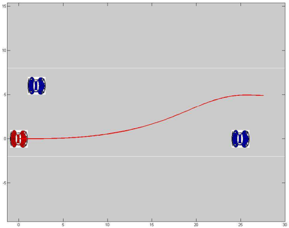
- compute the backward reachable set for such scenario;
- compute sensitivity analysis for CAS avoidance trajectory to study the influence of parameters on the CAS trajectory;
- compute a maximum sensor tolerance in initial data measurements such that the CAS avoidance trajectory is still admissible.

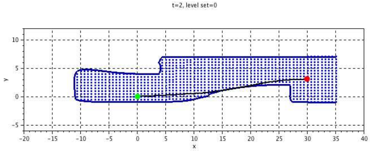
Comparison with CAS - Example of testing process

Input

Output of VTM

- verify if the starting point from which the CAS draws the optimal trajectory is in a safety area (**blue**) or in a collision area (white);

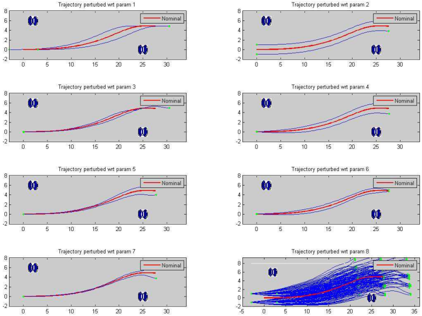




Comparison with collision by steering algorithm - Example of testing process

Output of VTM

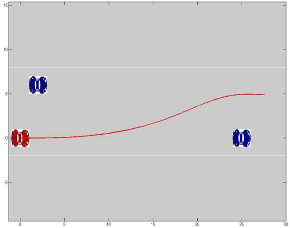
- verify if an avoidance optimal trajectory is still admissible also with sensor errors;
  States: xpos, ypos, vel, yaw angle, yaw angle vel, slip angle, steering angle.
  Controls: steering angle vel, braking force.



Comparison with collision by steering algorithm - Example of testing process

Output of VTM

- define r such that the optimal (**red**) trajectory is still admissible if sensor tolerance (in figure below is represented by r) is smaller than a value depending on the scenario and on the trajectory; we have that for this specific case:



The radius of a ball around all initial values is 0.07873597 meters

The radius of xpos is 9.32604041 meter

The radius of ypos is 0.49497475 meter

The radius of vel is 9.27426502 meter/sec

The radius of yaw angle is 0.11286939 rad

The radius of yaw angle vel is 1.27863692 rad/sec

The radius of slip angle is 0.19859987 rad

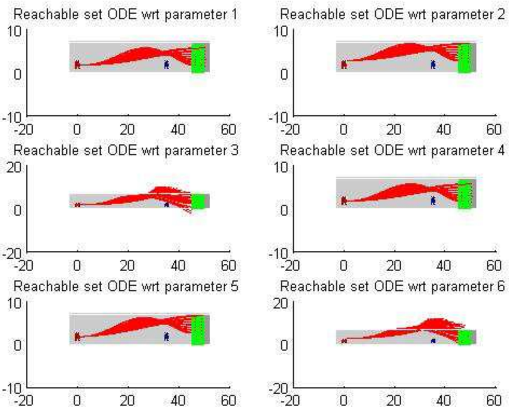
The radius of steering angle is 0.11178071 rad

more

Thanks for your attention!

- Questions?
- Further Information: ilaria.xausa@volkswagen.de

Acoustic warning



parameter1 = x position

parameter2 = y position

parameter3 = yaw angle

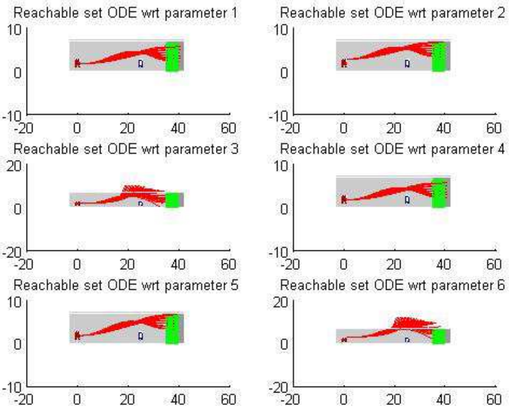
parameter4 = x velocity

parameter5 = y velocity

parameter6 = all parameters together

back

Braking warning



parameter1 = x position

parameter2 = y position

parameter3 = yaw angle

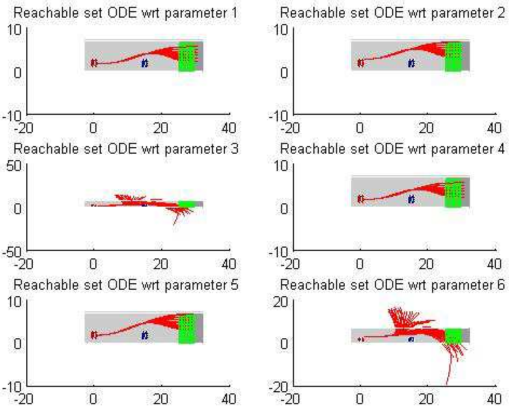
parameter4 = x velocity

parameter5 = y velocity

parameter6 = all parameters together

back

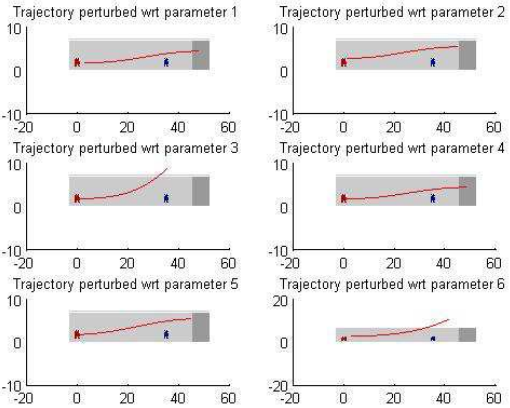
Automatic braking



parameter1	=	x position
parameter2	=	y position
parameter3	=	yaw angle
parameter4	=	x velocity
parameter5	=	y velocity
parameter6	=	all parameters together

back

Acoustic warning



parameter1 = x position

parameter2 = y position

parameter3 = yaw angle

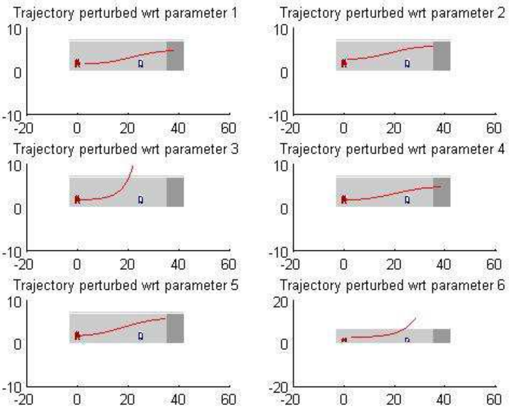
parameter4 = x velocity

parameter5 = y velocity

parameter6 = all parameters together

back

Braking warning



parameter1 = x position

parameter2 = y position

parameter3 = yaw angle

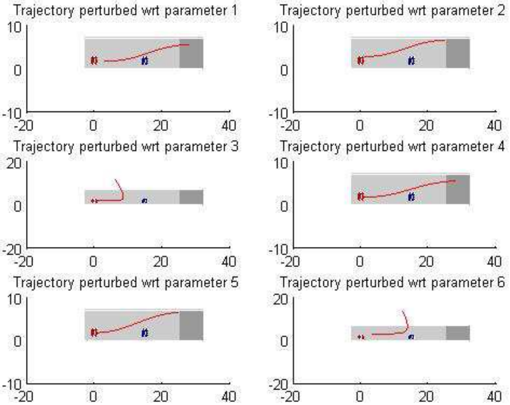
parameter4 = x velocity

parameter5 = y velocity

parameter6 = all parameters together

back

Automatic braking



parameter1 = x position

parameter2 = y position

parameter3 = yaw angle

parameter4 = x velocity

parameter5 = y velocity

parameter6 = all parameters together

back

ODE-Sensitivity

The ODE-Sensitivity of the state is defined as the partial derivative of the state mapping with respect to p for a fixed optimal control:

$$S(\cdot) := \frac{\partial z}{\partial p}(\hat{u}, \hat{p})(\cdot)$$

and it is given by solving the sensitivity differential equation

$$S'(t) = f'_z(\hat{z}(t), \hat{u}(t))S(t), \qquad S(0) = \frac{dz_0}{dp}(p).$$

Considering the approximation to the optimal perturbed trajectory

$$\begin{aligned} z(\hat{u}(p), p)(\cdot) \quad &\approx \quad \hat{z}(\cdot) + \frac{dz}{dp}(\hat{u}, \hat{p})(\cdot)(p - \hat{p}) \\ &= \quad \hat{z}(\cdot) + \frac{\partial z}{\partial p}(\hat{u}, \hat{p})(\cdot)(p - \hat{p}), \end{aligned}$$

for fixed optimal control $\hat{u}$, we can approximate trajectories for neighboring parameters.

back

FIACCO-Sensitivity

The Fiacco-Sensitivities of the state and the control are based on a parametric sensitivity analysis of the optimal solution:

$$\frac{dz}{dp}(\hat{u},\hat{p}) = \frac{\partial z}{\partial u}(\hat{u},\hat{p})\frac{d\hat{u}}{dp}(\hat{p}) + \frac{\partial z}{\partial p}(\hat{u},\hat{p}) \quad \text{and} \quad \frac{d\hat{u}}{dp}(\hat{p}).$$

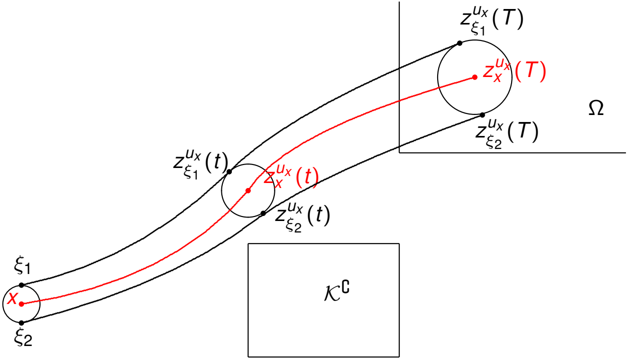
Such derivatives exist and can be computed using the linearized necessary Karush-Kuhn-Tucker conditions in an optimal solution $(\hat{z},\hat{u})$, as shown by A. V. Fiacco. Considering the approximation to the optimal perturbed trajectory

$$z(\hat{u}(p),p)(\cdot) \approx \hat{z}(\cdot) + \frac{dz}{dp}(\hat{u},\hat{p})(\cdot)(p-\hat{p}),$$

we can perform the perturbed trajectories.

back

Radius Estimation: Idea



back

Radius Estimation: Idea

$\bar{z} := z(\hat{u}, p)$ is the solution of the perturbed OCP with fixed optimal control $\hat{u} = u(\hat{p})$, and initial data $\bar{z}_0 = z_0(p) \in B(z_0, r)$ for some $r > 0$.
In first approximation we have (for r small)

$$z(\hat{u}, B(z_0, r))(\theta) \simeq \hat{z}(\theta) + rM_\theta B, \quad M_t := \exp\Big(\int_0^t D_z f(\hat{z}(s), \hat{u}(s))ds\Big)$$

Thus the state constraints are, in first approximation, equivalent to

$$\sup_{e \in B} \varphi(\hat{z}(T) + rM_T e) \leq 0, \text{ and } \sup_{\theta \in (0,T)} \sup_{e \in B} g(\hat{z}(\theta) + rM_\theta e) \leq 0$$

We derive the following sufficient condition:

$$\varphi(\hat{z}(T)) + r\|M_T^\perp \nabla\varphi(\hat{z}(T))\| \leq 0 \text{ and } \sup_{\theta \in (0,T)} g(\hat{z}(\theta)) + r\|M_\theta^\perp \nabla g(\hat{z}(\theta))\| \leq 0$$

and so:

$$r \leq \min\left(\frac{-\varphi(\hat{z}(T))}{\|M_T^\perp \nabla\varphi(\hat{z}(T))\|}, \frac{-g(\hat{z}(\theta))}{\sup_{\theta \in (0,T)} \|M_\theta^\perp \nabla g(\hat{z}(\theta))\|}\right)$$

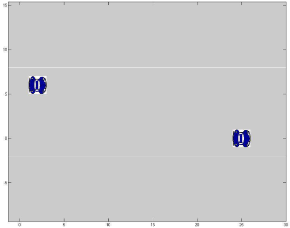
back

Comparison with collision by steering algorithm - Example of testing process

Input of VTM

Scenario:

- First obstacle: $x = 25$ [m], $y = 0$ [m], $v = 0$ [m/s], $\psi = 0$ [rad].
- Second obstacle: $x = 2$ [m], $y = 6$ [m], $v = 0$ [m/s], $\psi = 0$ [rad].



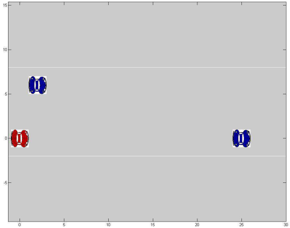
back

Comparison with collision by steering algorithm - Example of testing process

Input of VTM

Scenario:

- First obstacle: $x = 25$ [m], $y = 0$ [m], $v = 0$ [m/s], $\psi = 0$ [rad].
- Second obstacle: $x = 2$ [m], $y = 6$ [m], $v = 0$ [m/s], $\psi = 0$ [rad].
- Car: $x = 0$ [m], $y = 0$ [m], $v = 20$ [m/s], $\psi = 0$ [rad].



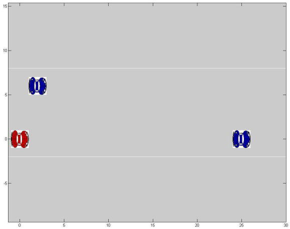
back

Comparison with collision by steering algorithm - Example of testing process

Input of VTM

Scenario:

- First obstacle: $x = 25$ [m], $y = 0$ [m], $v = 0$ [m/s], $\psi = 0$ [rad].
- Second obstacle: $x = 2$ [m], $y = 6$ [m], $v = 0$ [m/s], $\psi = 0$ [rad].
- Car: $x = 0$ [m], $y = 0$ [m], $v = 20$ [m/s], $\psi = 0$ [rad].



CAS output:

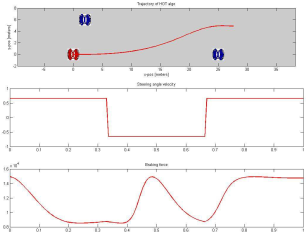


Figure : Plot of the trajectory (first row), with controls (steering velocity in second row and braking force in last row).

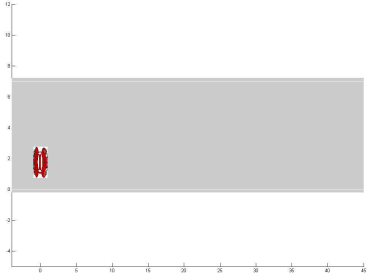
back

Comparison with collision by braking algorithm - Example of testing process

Input of VTM

Scenario:

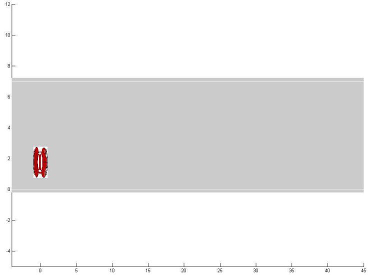
- Car: $x_C = 0$ [m], $y_C = 1.75$ [m], $v_C = 50$ [m/s], $\psi_C = 0$ [rad].



back

Comparison with collision by braking algorithm - Example of testing process

Input of VTM



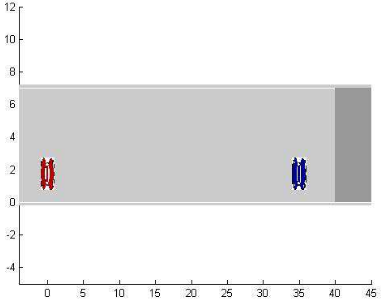
Scenario:

- Car: $x_C = 0$ [m], $y_C = 1.75$ [m], $v_C = 50$ [m/s], $\psi_C = 0$ [rad].
- Obstacle: x_O such that $\|x_O - x_C\| \in \{w_1, w_2, w_3\}$, $y_O = 1.75$ [m], $v_O = 0$ [m/s], $\psi_O = 0$ [rad].

back

Comparison with collision by braking algorithm - Example of testing process

Input of VTM



Scenario:

- Car: $x_C = 0$ [m], $y_C = 1.75$ [m], $v_C = 50$ [m/s], $\psi_C = 0$ [rad].
- Obstacle: x_O such that $\|x_O - x_C\| \in \{w_1, w_2, w_3\}$, $y_O = 1.75$ [m], $v_O = 0$ [m/s], $\psi_O = 0$ [rad].

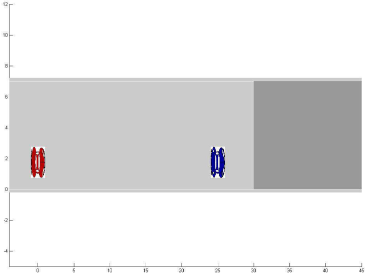
CAB output:

- **acoustic warning:** $w_1 \in [43.8889, 32.222]$, **and we take** $w_1 = 35$ **[m];**

back

Comparison with collision by braking algorithm - Example of testing process

Input of VTM



Scenario:

- Car: $x_C = 0$ [m], $y_C = 1.75$ [m], $v_C = 50$ [m/s], $\psi_C = 0$ [rad].
- Obstacle: x_O such that $\|x_O - x_C\| \in \{w_1, w_2, w_3\}$, $y_O = 1.75$ [m], $v_O = 0$ [m/s], $\psi_O = 0$ [rad].

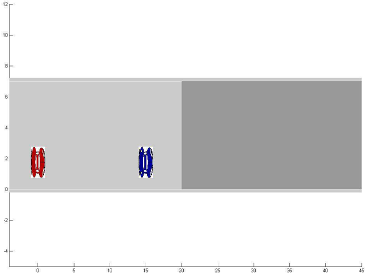
CAB output:

- **acoustic warning:** $w_1 \in [43.8889, 32.222]$, **and we take** $w_1 = 35$ **[m];**
- **braking warning:** $w_2 \in [31.6667, 20.5556]$, **and we take** $w_2 = 25$ **[m];**

back

Comparison with collision by braking algorithm - Example of testing process

Input of VTM



Scenario:

- Car: $x_C = 0$ [m], $y_C = 1.75$ [m], $v_C = 50$ [m/s], $\psi_C = 0$ [rad].
- Obstacle: x_O such that $\|x_O - x_C\| \in \{w_1, w_2, w_3\}$, $y_O = 1.75$ [m], $v_O = 0$ [m/s], $\psi_O = 0$ [rad].

CAB output:

- **acoustic warning:** $w_1 \in [43.8889, 32.222]$, **and we take** $w_1 = 35$ **[m];**
- **braking warning:** $w_2 \in [31.6667, 20.5556]$, **and we take** $w_2 = 25$ **[m];**
- **automatic braking:** $w_3 \in [19.4444, 2.22222]$, **and we take** $w_3 = 15$ **[m];**

back